International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Deployment of Virtual Resources Using Template Management Technology on Openstack

S. L. Mani Deepu[1], Hemavathi D[2]

[1]M. Tech Cloud Computing, Department of Information Technology, SRM University, Kattankulattur, Chennai, Inda

[2]Assistant Professor (Sr.G), Department of Information Technology, SRM University, Kattankulathur, Chennai , India

Abstract: ***This Paper actually focuses on the development of template management technology to build virtual resources environments on openstack. We have technologies which deploy a set of virtual resources based on system environmental templates to enable easy building expansion or migration of cloud resources. Openstack heat and Amazon cloud Formation are template deployment technologies and build stacks which are set of virtual resources based on templates. These existing technologies have some problems .Heat and CloudFormation transaction management of stack create or update are insufficient Heat and CloudFormation do not have sharing mechanism of templates .Heat Cannot extract templates from existing environments. Heat does not change reflect the actual environment changes to stack information. The goal of this to create a template management technology with necessary improvements. It has a mechanism of transaction management like rollback in case of abnormal failures during stack operations. It shares templates among end users and system integrators. It extracts templates from existing environments.***

Keywords: OpenStack, Template management server, Heat, Cloud Formation, Iaas

1. Introduction

Cloud computing technologies such as virtualization and scale-out have been progressed and many other providers have started cloud services. Iaas service provides hardware resources of cpu via network. openstack is one of the major opensource Iaas software and adoptions of opensource Iaas software is increasing. With a wide spread of cloud services, technologies which deploy a set of virtual resources based on system environmental templates to enable easy building, expansion, migration of cloud virtual resources have emerged. For example, openstack Heat and Amazon CloudFormation are template deployment technologies and build stacks which are sets of virtual resources based on templates. Heat and CloudFromation transaction managements of stack create and update are insufficient. Heat and CloudFormation do not have sharing mechanism of templates .Heat Cannot extract templates from existing virtual envionments. Heat does not reflect the actual environment changes such as virtual environment changes, such as virtual machine deletion by openstack nova API.

The openstack project is a platform for developing, deploying and hosting cloud computing solutions using open source software. OpenStack is an open source infrastructure as a service (IaaS) initiative for creating and managing large groups of virtual private servers in a cloud computing environment. The primary objective behind openstack project is to create a global standard and software stack for developing cloud solutions helping cloud providers and end-users alike. The project aims to build a unanimous cloud operating platform, where all the participating organizations will build cloud solutions that are not only scalable ,elastic and secure but also globally accessible OpenStack Heat, the orchestration service that allows you to spin up multiple instances, logical networks, and other cloud services in an automated fashion. some basic terminology.

- **Stack**: In Heat parlance, a stack is the collection of objects that will be created by Heat. This might include instances (VMs), networks, subnets, routers, ports, router interfaces, security groups, security group rules, auto-scaling rules, etc.
- **Template**: Heat uses the idea of a template to define a stack. If you wanted to have a stack that created two instances connected by a private network, then your template would contain the definitions for two instances, a network, a subnet, and two network ports. Since templates are central to how Heat operates.
- **Parameters:** A Heat template has three major sections, and one of the sections defines the templates parameters. These are tidbits of information-like a specific image ID,or particular network ID-that are passed to the Heat template by user. This allows us to create more generic templates that could potentially use different resources.
- **Resources:** Resources are the specific objects that Heat will create and/or modify as part of its operation, and the second of the three major sections in a Heat template.
- **Output:** The third and last major sections of a Heat template is the output, which is information that is passed to the user ,either via openstack Dashboard or via the heat stack-list and het stack-show commands.
- **HOT:** Short for Heat Orchestration Template .HOT is one of two template formats used by Heat. HOT is not backwards-compatible with AWS CloudFormation templates and can only be used with openstack. Templates in HOT format are typically but not necessarily expressed as YAML.
- **CFN:** Short for AWS CloudFormation, this is the second template format that is supported by Heat.CFN-formatted templates are typically expressed in JSON.

Architecturally, Heat has few major components:

- The heat-api component implements an openstack-native RESTful API.This components processes API requests by sending them to the Heat engine via AMQP.

Licensed Under Creative Commons Attribution CC BY

- The heat-api-cfn component provides an API compatible with AWS CloudFormation and also forwards API requests to the Heat engine over AMQP.
- The heat-engine component provides the main orchestration functionally

All of these components would typically be installed on an openstack "controller" node that also housed the API servers for Nova, Glance, Neutron, etc. Heat uses a back-end database for maintaining state information.

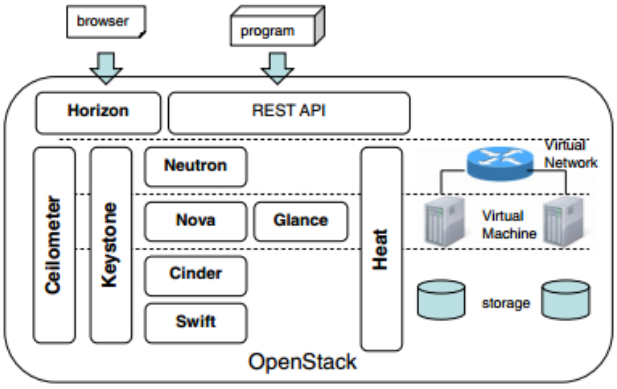

Figure 1: A Basic Implementation of the Concept

2. Outline of openstack

Openstack, CloudStack and Eucalyptus are major opensource Iaas software, and among them openstack community is active because many providers contribute developments and adopted services rapidly increasing. openstack is composed of the function blocks which manage logical/virtual resources deployed on physical resourcs, the function block which provides single sign on authentication among other function blocks and the function block which orchestrates a set of virtual resources. Neutron controls virtual networks.OVS(Open Virtual Switch) and other software switches can be used as a virtual switch. Nova controls virtual machines KVM(Kernel virtual machine) and others can be used as hypervisors of VMs.Cinder manages the block storages and can attach a logical volume to a VM like a local disk.Swift manages object storages Glance manages Image files. Keystone is a base which performs single sign on authentication of these function blocks. Heat is an orchestration deployment function to create or update virtual resources instances using Nova, Cinder or other blocks based on a text template.celiometer is a metering service of virtual resource usage. The functions of openstack are used through REST(Representation State Transfer) APIs.There is also web GUI called Horizon to use the functions of openstack.

Problems in Existing Template Technologies

Openstack Heat and CloudFormation are technologies which deploy a set of virtual resource instances based on templates which contain information of virtual resource environment and are described by JSON,YAML or other text format .Both call a set of virtul instances which are deployed based on template "stack" and provide API's to operate stacks. However, these API's provide primitive CRUD (Create, Read, Update, Delete) operations of stack and there are some insufficient points for business use.

OpenStack Architecture

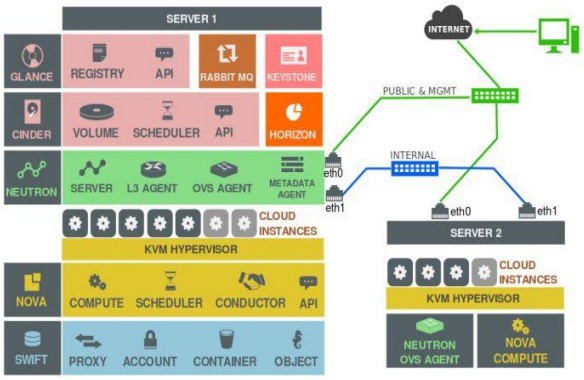

Figure 2: OpenStack Architecture (Single Node)

OpenStack version used here for the Design of the project is Juno Update on top of Ubuntu 14.04(Trusty Tahr).[3]. OpenStack is a collection of open source software projects that enterprises/service providers can use to setup and run their cloud compute and storage infrastructure. Rackspace and NASA are the key initial contributors to the stack. Rackspace contributed their "Cloud Files" platform (code) to power the Object Storage part of the OpenStack, while NASA contributed their "Nebula"platform (code) to power the Compute part. OpenStack consortium has managed to have more than 150 members including Canonical, Dell, and Citrix etc. Heat is the main project in the openstack orchestration program. It implements an orchestration engine to launch multiple composite cloud applications based on template in the form of text files that can be treated like code. A native template format is evolving, but Heat also endeavours to provide compatibility with the AWS CloudFormation templae format, so that many existing CloudFormation templates can be launched on openstack. Heat also provides an autoscaling service that integrates with ceilometer, so you can include a scaling group as a resource in a template.

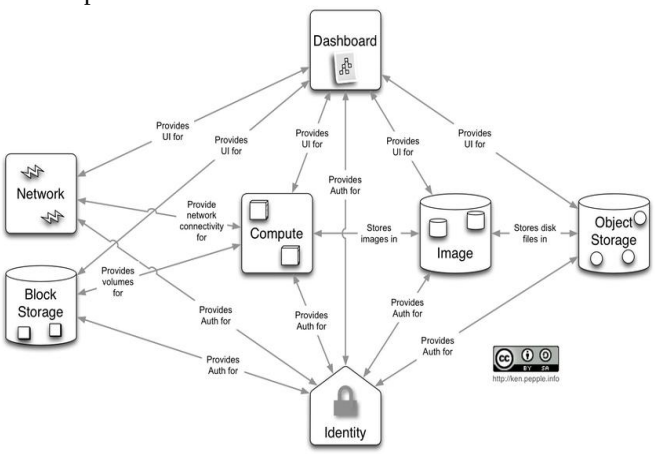

Figure 3: Data Flow in OpenStack

There are main service families in OpenStack Juno:

- Nova - Compute Service
- Swift - Storage Service
- Glance - Imaging Service
- Keystone - Identity Service
- Neutron - Networking service
- Cinder - Volume Service

Licensed Under Creative Commons Attribution CC BY

- Horizon - Web UI Service
- Heat – Orchestration.

We will see Heat that is intended for this paper.

Orchestration multiple composite cloud applications by using either the native HOT template format or the AWS Cloud Formation template format,through both an Openstack-native REST API and a Cloud Formation-compatible Query API.
Functions and Features:

- Instance life cycle management
- Management of compute resources
- Networking and Authorization
- REST-based API
- Asynchronous eventually consistent communication
- Hypervisor agnostic: support for Xen, XenServer/XCP, KVM, UML, VMware vSphere and Hyper-V.

Modules

a) Transaction management of stack create, update and delete.
b) Sharing of templates.
c) Extracting the templates from existing environment.
d) Actual environment changes reflecting to stack information.

3. Transaction Management of Stack Create, Update and Delete

Heat and CloudFormation transaction managements of stack create, update and delete are insufficient and may end up with a half-finished stack processing. It is not acceptable for some business users because some resources failures may lead critical problems. For example, if a VM creation is successful but a logical router security setting is failed, the VM may have a risk of abuse. Therefore, when we create a stack, it is necessary to delete and roll back all resources in case of any failures of resource creation. And when we delete a stack, it is necessary to retry and delete all resources in case of any failures of resource deletion.

When we update a stack, orchestration functions check a difference between previous template and new template, then create, delete or update resources to fill up the difference. Specifically, a resource which is in previous template and is not in new template is deleted, a resource which is not in previous template and is in new template is created, a resource which is changed from previous template to new template is created after deletion or updated.

In OpenStack, some resources can be updated but some cannot be. (e.g. network connection change can be updated but VM RAM size change needs to delete VM once, then create new one). In stack update case, individual resource creation, deletion and update may be operated, so that there may be a case we cannot roll back all operations. For example, a volume is deleted successfully then a VM creation is failed, we cannot roll back because the volume is already deleted.

Therefore, in stack update case, the TM server tries to roll back or roll forward for each OpenStack API transaction (not all API transactions) when each OpenStack API processing is failed. The function of stack transaction management can be generalized for deployment management with multiple resources.

Resource deployment needs Create, Update and Delete transaction managements and also needs a valid order of each resource operation. Our proposal of stack transactions in case of failure and orders of each resource operation can be used also CloudStack, Eucalyptus and other Cloud platforms multiple resources provisioning because a virtual resource dependence (e.g. VM needs at least one volume) is almost same in IaaS platforms.To follow these policies, the stack operation function of TM server manages orders of OpenStack API calls and those transactions.

Most of OpenStack APIs are asynchronous, the TM server retries a API or calls a purge API, or reverse API to decide state of OpenStack resource when a API transaction is failed. Note that a reverse API means the reverse process of each API (e.g. volume deletion API is a reverse API for volume creation). In this way, we can prevent a half-way state of stack during stack operations., we can guarantee the precondition of resource creation in stack create case (e.g. a VM needs at least one volume).

Template sharing

CloudFormation and Heat do not have a mechanism of template sharing. Our TM server provides a function to share templates and facilitate templates re-use. For example, when a small business owner would like to build a shopping site, a System Integrator provides a verified Web 3-tier structure template, then the small business owner selects the template and build the environment with one or two clicks. If we share templates unconditionally, there is a risk of malicious template spreading. Thus, it is necessary to limit a range of template sharing within contractual relationships. Here, we explain logics of template sharing.

There are two methods to register a template: template extraction and template upload. The function described in Template extraction from existing tenant extracts a valid template in an extraction case. In the other side, the template sharing function validates a template in a template upload case because a template described by a user may have format or logical errors. – Each template creator can set a scope of disclosure for each template. There are 3 options for disclosure; only the creator-self, all users who have contract with the creator and users selected by the creator. Service providers or System Integrators can share templates to subordinate users by setting a scope of disclosure. If System Integrators have multiple tier contractual relationships, there are 2nd tier subordinate users. In this case, 1st tier subordinate System Integrator downloads a template of upper tier System Integrator and registers it as its template. This is to restrict scope of disclosure within direct contractual relationships. Because each System Integrator prefers to sell its own brand, the template sharing function conceals templates upper than two tiers. Figure 3 shows an image of template sharing relations.

Licensed Under Creative Commons Attribution CC BY

Template Extraction from Existing Environment:

Heat main targets are operations of stacks and Heat cannot extract a template from non-stack environment. And there is a restriction that each virtual resource belongs to only one stack. Based on them, we propose logics to extract a template from an existing tenant. – We extract whole virtual resources on an existing tenant to a template. If there is unnecessary resource in an extracted template, a user edits the template after downloading. This is because there is no stack that we cannot restrict corresponding resources for extraction. – Target resources to extract are volumes, virtual Layer 2 networks, VMs, logical routers and logical load balancers. Floating IPs are IP address resources that relate logical routers which connect the Internet. The Internet connected resources are shared by multiple stacks in general and VMs or logical load balancers which are assigned floating IPs may be shared by multiple stacks. Because a virtual resource only belongs to one stack, we do not include a floating IP to a template in extraction phase. Users can assign floating IPs after stack creation based on the extracted template. In the same way, shared virtual Layer 2 networks or logical routers (e.g. VPN connected routers) are out of scope for extraction because those are used by multiple stacks. – When users extract a template, they also can select whether to acquire images from volumes of tenant or not. When users create a stack, these images are used to replicate volumes. – During template extraction time, we block virtual resources operations in the tenant to prevent a change of target resources for extraction. – Extracted templates are held in the template sharing function described in Template sharing. Extracted templates can be used for stack create, update or download to edit. In this way, we can extract a template from an existing tenant and replicate an environment easily. The function of template extraction extracts a template from an existing environment. Our implementation extracts JSON or HOT template and the extracted template can be deployed both by Amazon CloudFormation and OpenStack Heat because a template is abstract text information and does not depend on IaaS platform. To generalize and adapt extracted template format to other IaaS platforms, we can use this function for Cloud migration to another platform or Cloud federation on Plural platforms.

4. Reflection of Environment Change to Stack Information

In case of stack update, orchestration functions check a difference between previous template and new template, then create, delete or update virtual resources to fill up the difference. However, Heat can only recognize an environment change by Heat API and does not know actual environment status.

a) Stack creates, update, delete by Heat Stack API

b) Individual resource creates, update, delete by other OpenStack API (Nova, Cinder and so on)

c) Resource deletion by user's manual operation. (e.g. VM shutdown via console)

d) Resource deletion by unintentional physical or virtual server down.

We do not have to care it because templates of Stack API are matched to actual environments after API process. Regarding to b), when users call OpenStack API (not Heat API), the TM server hooks the requests as OpenStack API proxy and reflects the environment change to stack information. If proxy model is difficult, the TM server may poll OpenStack DB to confirm environment changes. But each OpenStack API does not have a parameter of stack ID so that individual resource creation is not reflected to stack information. If a user would like to add a resource to a stack, a user needs to call Heat stack update API including the resource. Regarding to c), main case is a VM shutdown by user's manual operation. VM is operated by Libvirt on KVM. Therefore, the TM server can reflect the VM down status to stack information by receiving notifications of Libvirt or other monitoring agents. Regarding to d), the TM server reflects resources down to stack information by receiving a notification of each resource monitoring agent like c) case. Based on reflected stack information on a)-d), the TM server can update or delete stacks as users expect. Table 1 shows a comparison of reflection of actual environment change to stack information in Havana Heat case and our TM server case. Havana Heat updates environment changes to stack information only in Stack API use. Our TM updates environment changes to it except for resource create by individual OpenStack API call. The function of environment change reflection function is generalized to a difference resolve function of actual environment and management layer. The difference has a problem not only in a stack update case but also in an individual resource provisioning case. For example, OpenStack Neutron manages a resource state in OpenStack DB and does not care an actual completion of resource provisioning after it has written the requests to DB. Thus, there is a possibility that a VM is active but access control setting of a logical router to the VM is not available. Because a Cloud provider business is charging fees for provisioned resources, it is fatal to charge a resource not created yet. The function can be used for resolving these differences to collect actual environment information by monitoring modules such as Libvirt and Pacemaker or by hooking requests as a proxy.

5. Design Coding

Templates:

```
"Resources":{
 "vm-1" : {
 "Type": "os:: Nova::server";
 "properties":{
 "networks": {{
 "port" : {"Ref" : "vm-1-port"}
 }]
"availability_zone" : {"Ref" : "vm-1-az"},
"description": { "Ref": "vm-1-description"},
 "name" :{"Ref": "vm-1-name"},
 "flavor" : {"Ref":"vm-1-flavour"},
 "image" : {"Ref":"vm-1-image"},
"block_device_mapping" : [{
 "volume _id": {Ref": "volume-a"},
 "delete_on_termination" : "0"
},{
 "volume_id": {"Ref" : "volume-b"},
```


Licensed Under Creative Commons Attribution CC BY

"delete_on_termination":"0"
}]
}
}
}
"Resources":{
"vm-1" : {
"Type": "os:: Nova::server";
"properties":{
"networks": {{
"port" : {"Ref" : "vm-2-port"}
}]
"availability_zone" : {"Ref" : "vm-2-az"},
"description": { "Ref": "vm-2-description"},
"name" :{"Ref": "vm-2-name"},
"flavor" : {"Ref":"vm-2-flavour"},
"image" : {"Ref":"vm-2-image"},
"block_device_mapping" : [{
"volume _id": {Ref": "volume-a"},
"delete_on_termination" : "0"
},{
"volume_id": {"Ref" : "volume-b"},
"delete_on_termination":"0"
}]
}
}
}

6. Conclusion

The above implemented template management technology for end users to build virtual environments on openstack. Template management server prevented half finished stack because it rolled back all operations in case of abnormal failure of stack update. Template sharing to end users who have contracts to system integrators or providers can replicate virtual resource environments on new tenants easily. our server extract a template from non-stack environment except shared resources. Users could update stack as expected because our server reflected actual environment change such as VM shutdown or other openstack API operation to stack information.

7. Future Work

Improving the TM server for openstack new versions.Juno is the new version of openstack and provides new functions to catch up Amazon Web Services nd also improving the software softare quality of TM server and verify the feasibility of existing OSS(operations support system) interconnections to provide production carrier Iaas services based on openstack.

References

[1] Deployment of template management technology for easy deployment of virtual resources on openstack.yoji yamato,mashahito muroi,kentaro
[2] Amazon Elastic compute cloud website http://aws.amazon.com/ec2/
[3] Opensttack web site.http://www.openstack.org/
[4] Cloudstack website http://www.cloudstackapache.org/
[5] Amazon Cloud Formation website.http ://aws.amazon/cloudformation/
[6] .Nurmi D,Wokshi R, Grzegorczyk C,obertelli G,Soman S,YOuseff L,Zagorodnov D(2009) The Eucalyptus Opensource Cloud Computing system.IN proceedings of cluster computing and the Grid, 2009.
[7] RightScale Server Templates website,http://wwww.rightcale.com/blog/Cloud-management-best-practices/rightscale-servertemplates-explained.

Licensed Under Creative Commons Attribution CC BY